REMARKS

The Office Action of January 29, 2007 has been reviewed and the comments therein were carefully considered. Claims 1-17 are pending. Claims 1-17 stand rejected.

Claim Rejections Under 35 USC §102

Claims 1-17 are rejected under 35 USC §102(e) as being clearly anticipated by Kasichainula, et al., U.S. Patent No. 6,941,561. Applicants respectfully traverse the rejections.

Kasichainula discloses a method of distributing a program written using object orientated programming so that portions of the written program may be executed on more than one computer across a network. Two proxy objects are generated dynamically to enable communication over the intervening network by intercepting calls. The proxy objects allow method calls written for local invocation to be invoked over a network. The two proxies cooperate to hide the fact that the objects actually reside on different machines from the programmer, thereby sparing the programmer any need to be aware of the distributed nature of the system when writing his code.

Independent claims 1 and 13 include the claimed feature of "wrapping a reference to a second object within a second context with a proxy wrapper." The Office Action states and Applicants agree that Kasichainula does not disclose "wrapping" a reference. However, the Office Action considers the claimed feature of "wrapping a reference to a second object with a second context with a proxy wrapper" to be inherently provided by Kasichainula. In particular, the Office Action states that in Kasichainula "code is not changed to enable access to remote objects and the programmer is unaware of the change that occurs to enable access objects as if they were local (i.e. indirectly)." (Office Action, page 4). Applicants respectfully disagree with the Office Actions position. Applicants respectfully submit that Kasichainula does not disclose the claimed feature of "wrapping . . . with a proxy wrapper."

In Kasichainula, <u>two proxy objects</u> are generated dynamically and used to enable communication over an intervening network by intercepting calls. In particular, Kasichainula's specification at col. 5 lines 5-11 states "When the call returns a data stream, Y" 406 and Y' 405 cooperate to communicate the data that the data stream represents across the network and present it to the caller as if it were local data." However, the use of two proxy objects does not disclose the claimed feature of "<u>wrapping a reference</u> to a second object within a second context with <u>a</u>

proxy wrapper. (Emphasis added). Therefore, for at least this reason independent claims 1 and 13 are distinguishable over Kasichainula.

Independent claims 1 and 13 are allowable for at least an additional reason. Claims 1 and 13 include the claimed feature of "the first context defining at least a first set of arbitrary invariants on a first set of arbitrary objects including the first object." The Office Action cites Kasichainula at col. 8 lines 26-31 for support of this claimed feature. Col. 8 lines 26-31 of Kasichainula states:

> [O]bject Y 503 may update a counter in its copy of object Z. Before the copy of object Z is returned to machine 501, another object in machine 501 may update the same counter in the actual object Z residing on machine 501. When object Y on machine 509 completes its operation on its copy of object Z, the object must be recopied to machine 501. . . .

Applicants respectfully submit that the cited portion of Kasichainula does not even remotely disclose defining a set of arbitrary invariants. In fact, the Office Action does not even mention the claimed "arbitrary invariants." Therefore, for at least this reason, Applicants respectfully submit that independent claims 1 and 13 are in condition for allowance. Dependent claims 2-5 and 14-17 are in condition for allowance for at least the same reason as independent claims 1 and 13 from which they ultimately depend.

Independent claim 6 includes the claimed feature of "at least one first object communicates with any of the at least one second object via indirect references wrapped in proxy wrappers." (Emphasis added). Applicants respectfully submit that a proxy wrapper which includes indirect references may not properly be equated with the creation of two proxy objects in Kasichainula. Therefore, for at least this reason Applicants respectfully submit that independent claim 6 is in condition for allowance. Dependent claims 7-11 which ultimately depend from independent claim 6 are in condition for allowance for at least the same reason as independent claim 6.

Independent claim 12 includes the claimed feature of "at least one agile object . . . in that the at least one agile object have no permanent context . . . ." With respect to independent claim 12, the Office Action states "the proxy objects are considered to provide the feature; since they are not fixed (generated dynamically via the abstract) to enable access as if they were local."

Applicants respectfully traverse the rejection. In an embodiment, Applicant's specification states on page 12-13:

> To a caller object within the context 200, the agile object 214 executes within the context 200 – that is, current execution thread for the agile object 214 is executes within the context 200 – such that to the caller object within the context 200 the agile object 214 appears as if it were context-bound in the context 200 for communication therewith. Likewise, to a caller object within the context 202, the agile object 214 executes within the context 202, such that to the caller object within the context 202 the agile object 214 appears as if it were context-bound in the context-bound in the context 202. The agile object 214 can directly access any other object in the context in which it is executing.

Applicants respectfully submit that an agile object is able to execute within multiple contexts. Kasichainula does not disclose objects that may execute within multiple contexts. Therefore, for at least this reason, Applicant respectfully submits that claim 12 is distinguishable over Kasichainula.

Claims 1-17 are rejected under 35 USC §102(e) as being clearly anticipated by Montgomery, (NPL reference entitled Sunsoft's Object Lesson). Applicants respectfully traverse the rejections.

Montgomery discloses a method the same source (and object) code is used to access objects within the same process, within the same machine but in different processes, within a network of cooperating machines.

Independent claims 1 and 13 include the claimed feature of "wrapping a reference to a second object within a second context with a proxy wrapper." The Office Action submits that Montgomery teaches a proxy wrapper by passing objects through a client stub. Montgomery states:

> stub marshals the argument – places them into a communication buffer with enough information so that the server can figure out what's going on. The stub then passes them to the subcontract which, in turn, executes the call to the server based object.

Applicants respectfully submit that marshalling an object into a communication buffer is not wrapping a reference to a second object within a second context with a proxy wrapper. Therefore, for at least this reason, Applicants respectfully submit that independent claims 1 and 13 are in condition for allowance. Dependent claims 2-5 and 14-17 are in condition for allowance for at least the same reason as independent claims 1 and 13 from which they ultimately depend.

Independent claim 12 includes the claimed feature of "at least one agile object . . . in that the at least one agile object have no permanent context . . . ." The Office Action submits that Montgomery discloses this claimed feature. The cited portion of Montgomery states:

> Spring's memory-management system is fairly simple. A client talks to an address space object. The address space maps to a memory object – e.g. a file. These two objects are the result of the cooperation of the virtual-memory manager (VMM) and an external pager.

The two objects mapping to each other in Montgomery do not disclose or suggest an agile object that has no permanent context. Therefore, for at least this reason, Applicants respectfully submit that independent claim 12 is distinguishable over Montgomery.

Independent claim 6 includes the claimed feature of "at least one first object communicates with any of the at least one second object via <u>indirect references wrapped in proxy wrappers</u>." (Emphasis added). Applicants respectfully submit Montgomery does not disclose or suggest this claimed feature. Moreover, the Office Action does not specifically state were in Montgomery this claimed feature may be allegedly found. Therefore, for at least this reason, Applicants respectfully submits that independent claim 6 is in condition for allowance. Dependent claims 7-11 which ultimately depend from independent claim 6 are in condition for allowance for at least the same reason as independent claim 6.

Applicants respectfully request consideration of the pending claims and a finding of their allowability. A notice to this effect is respectfully requested. Please feel free to contact the undersigned should any questions arise with respect to this case that may be addressed by telephone.

Respectfully submitted,

Dated: March 29, 2007

By: William J. Allen 51,393
William J. Allen
Registration No. 51,393
Banner & Witcoff, Ltd.
10 South Wacker Drive
Suite 3000
Chicago, IL 60606
Tel: 312-463-5000
Fax: 312-463-5001